Cahier des Charges

Outil d'indexation de fichier

Par : Stéphane Wong, Enis Suljovic, Pierre Dargham, Marc Richard, Victor Oudin & Ali Ben El Hadj

01/04/2013

Introduction 2
Specifiaction 3
Objectif 3
Besoin non-fonctionnels ....
Besoin du produit 4
Utilisabilité 4
Simplicité 5
Portabilité 5
Besoins fonctionnels 6
Affichage du résultat 13
Use Case : 20
....
Cas 1: 21
Scénario nominal: 21
Post-conditions: 21
Cas 2: 22
Scénario nominal: 22
Post-conditions: 22
Cas 3: 23
Scénario nominal: 23
Post-conditions: 23
Besoin organisationnel 24
Développement de l'application 24
Livraison 24
Implantation 24
Standard 24
Besoin externe 25
Interopérabilité 25
Éthique 25
Légalité 25
Respect de la vie privée 25
Annexes 27

Introduction

Dans le cadre de l'unité de l'enseignement Génie Logiciel et afin d'avoir un aperçu du métier d'ingénieur en informatique, nous avons l'occasion de réaliser un projet informatique qui reprend le cadre de ce fait en entreprise. Celui-ci se déroule en groupe et implique une démarche préalable avant la réalisation concrète du projet. Nous allons donc présenter le cahier des charges ci-dessous, les spécifications du projet, le codage à proprement parler puis la livraison finale du projet avec les tests du logiciel devant le client.

Spécification

Objectif

Sur vos disques durs se trouvent des centaines de giga-octets de fichiers contenant des images du code des vidéos, courriers, etc. Pour trouver un fichier contenant une information bien précise, par exemple, les notes de cours, vous devez soit être très bien organisé, soit lancer un outils de recherche sur votre disque, par exemple « find » sur le système linux. Malheureusement, ces outils sont assez lourd si vous avez à balayer tout le disque.

Un système d'indexation et de recherche de fichier pallie cette lourdeur et permet de retrouver quasi instantanément les fichiers correspondant aux critères recherchés. Ces critères, aussi nommé « méta-donnée », peuvent être le nom du fichier, le contenu du fichier, le format du fichier, la date de sa dernière modification, ou toute information significative pour le fichier en question. Le résultat de la recherche est affiché très rapidement grâce a une indexation des fichiers du disque effectuée au préalable. Plusieurs systèmes d'indexation existent, le plus connu Google Desktop, qui est une adaptation du moteur de recherche à un système de fichiers local, indépendant du système d'exploitation. Les systèmes propriétaire (MAC OSX ou Windows) disposent de leur propre outils d'indexation. GNOME a également développé le système d'indexation Tracker qui est un logiciel libre. D'autres logiciels libres efficaces, comme Recoll, basé sur le moteur d'indexation Xapian, sont disponibles.

Les performances d'un tel système peuvent être mesurées selon différents critères : les formats de fichiers dont le contenu est examiné, la charge supplémentaire générée par la tâche de fond de pré-indexation, la complexité des requêtes encodant les critères de recherche, la portabilité sur différents système de fichiers, etc.

Besoin du produit

Utilisabilité

La qualité de l'ergonomie sera un facteur essentiel, étant donnée l'utilisation intensive qui sera faite de l'application.
Un fichier d'aide à l'utilisateur, présentant l'interface et les fonctionnalités sera disponible.
Une fois le logiciel démarré, l'utilisateur devra pouvoir lancer une recherche en 2 clics au plus.

Vitesse d'exécution

L'outil de recherche « *find* » sur le système linux est assez lourd lorsqu' il doit parcourir un disque entier. À contrario, notre système d'indexation et de recherche palliera cette lourdeur et permettra de retrouver quasi instantanément les fichiers correspondant aux critères recherchés.
Un premier résultat de recherche doit être fourni au maximum en 10 secondes, et l'ensemble des résultats de la recherche doit être donné en moins d'une minute.
L'application devra utiliser des algorithmes rapides et efficaces, dont un algorithme de Stemming ainsi qu'un algorithme d'indexation inversée.
Le temps d'exécution de la première indexation du logiciel ne devra pas dépasser dix minutes pour un espace de stockage raisonnable. On visera une vitesse d'indexation d'un maximum de 10 minute pour 100go de données.

Consommation en espace

L'outil d'indexation devra remplir une base de données de façon intelligente et devra être la plus condensée possible. Par exemple, 10Go de données ne devra pas créer une base de données d'une dizaine de Giga-octets. La taille de l'index ne devra pas dépasser 10% de la taille des fichiers indexés.

Recherche

L'utilisateur aura la possibilité de restreindre la recherche à certains types de fichiers, à un emplacement donné, au titre et / ou au contenu des fichiers. La recherche sera faite par terme, avec la possibilité d'utiliser des expressions logiques comme « AND » et « OR ».
Une recherche vocale est également disponible et s'active à l'aide d'un bouton.

Gestion des tâches

Le système d'indexation est composé de quatre parties :

1) le moteur d'indexation qui crée et met à jour l'index des fichiers
2) la base de donnée contenant l'index des fichiers
3) le moteur de recherche qui assure l'interface entre la base de données et l'utilisateur
4) Le daemon est lancé en arrière-plan et envoie les informations de modification au moteur d'indexation

Simplicité

Un utilisateur non averti devra pouvoir utiliser cette application sans devoir faire appel à une commande sur son shell. Une interface graphique intuitive sera donc disponible. Une recherche basique doit pouvoir être effectuée sans lecture du manuel d'utilisation.

Portabilité

Le logiciel devra fonctionner sur des systèmes de base UNIX ou GNU/Linux, un portage sur le système Windows n'est pour le moment pas considéré.

Besoins fonctionnels

L'application doit permettre :

> Un accès quasi immédiat à la liste des fichiers correspondant aux requêtes données par le client

> Les requêtes peuvent être le nom du fichier, dont le contenu, son format, la date de sa dernière modification ou toute autre information significative

> La pré-indexation des fichiers du disque sera effectuée en tâche de fond.

> Le système d'indexation est composé de quatre modules:

1° le moteur d'indexation qui est exécuté en tache de fond,

2° la base d'indexation contenant les résultats du moteur d'indexation,

3° le moteur de recherche qui assure l'interface entre la base de données et l'utilisateur

4° le daemon reste en arrière-plan et transmet au moteur d’indexation les modifications apportés sur les fichiers indexés ou non.

Interface utilisateur :

Ceci est la page d'accueil au démarrage du logiciel.
L'utilisateur peut, par le biais du client graphique, lancer une recherche et l'affiner grâce à plusieurs dates, ainsi que par le type de fichier à rechercher.
Plusieurs types peuvent être sélectionnés en même temps.



Une recherche simple

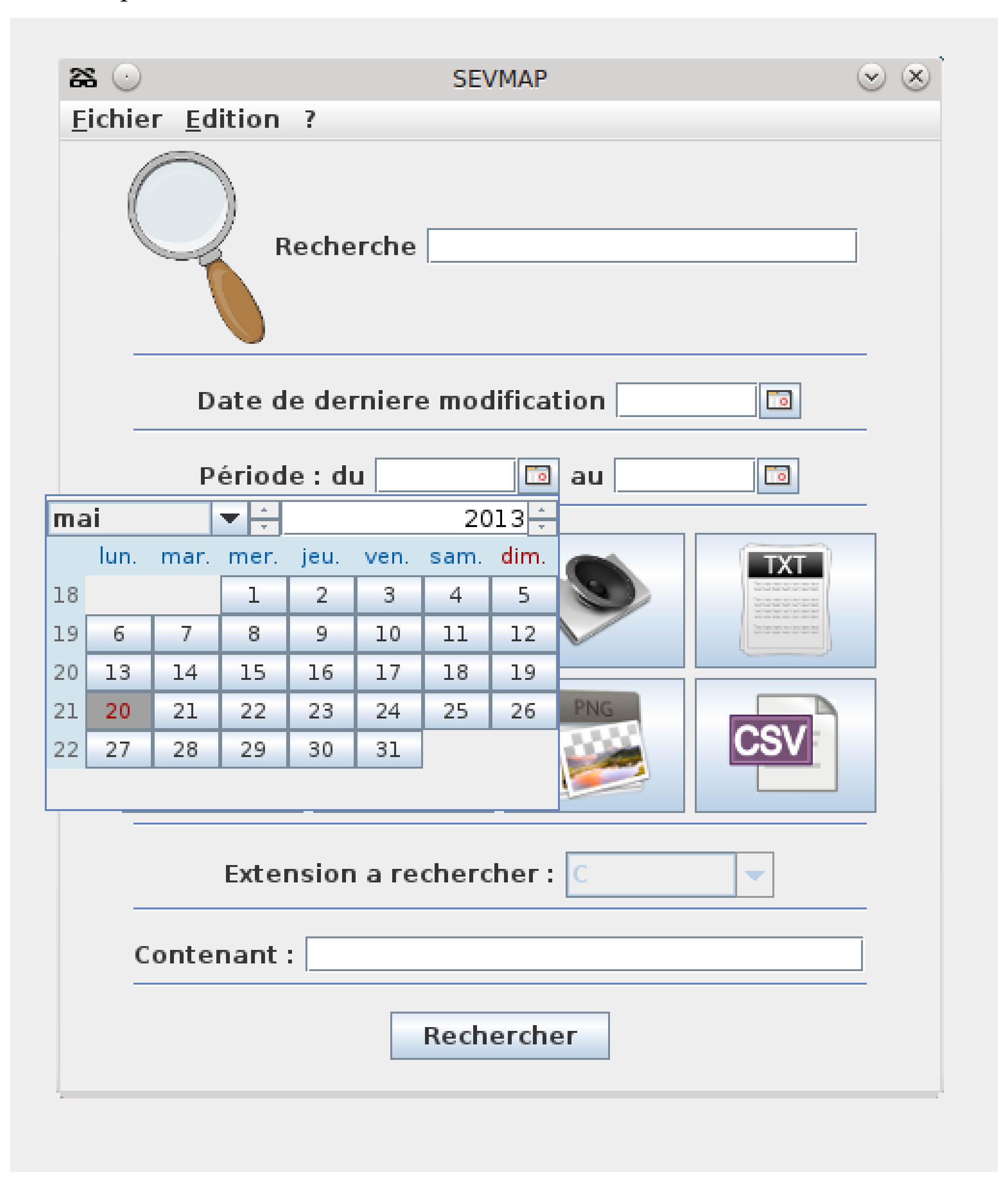
Une recherche basique peut être effectuée dans la barre de recherche comme suit :



Une recherche avancée peut être effectuée à l'aide des « **paramètres avancés** ». Le choix du paramètre est effectué par un clic sur l'icône. Celui-ci est alors surligné pour informer l'utilisateur que cela a bien été pris en compte:

Une recherche à l'aide du calendrier interactif

La sélection du temps pourra se faire par l'utilisation du petit calendrier interactif de ce type, après avoir cliqué sur l'icône.



Il faut tout de même préciser que chacun des champs du formulaire aura une longueur maximum à respecter. Celle-ci sera de l'ordre d'une centaine de caractères, et précisée au sein de l'interface. Si l'utilisateur tente d'entrer une chaine de caractères trop longue, il pourra constater que sa chaine sera tronquée par l'interface.

Une recherche par extension

Une option supplémentaire est offerte à l'utilisateur pour rendre la recherche plus pertinente.
Il s'agit de recherche par extension.
Etant donnée la multitude d'extension existante, il est préférable de se restreindre à certains types de fichier. Voici comment il faut procéder pour paramétrer ce type de recherche :



En sélectionnant l'une des icônes présentes au centre de l'interface, une boite de dialogue s'ouvrira, et il sera possible de définir les extensions à chercher selon la catégorie.
Une autre manière d'y parvenir est d'aller dans l'onglet 'Edition', puis de sélectionner 'Modifier les extensions'.

Voici la boite de dialogue correspondant :



Dans l'onglet Edition, il est également possible de supprimer les fichiers temporaires.
Ces fichiers temporaires sont en fait des liens symboliques des résultats de recherches précédentes.
En activant le bouton moniteur, situé en bas à gauche de la liste des boutons, on peut activer/désactiver la sélection d'extension à rechercher comme ceci :



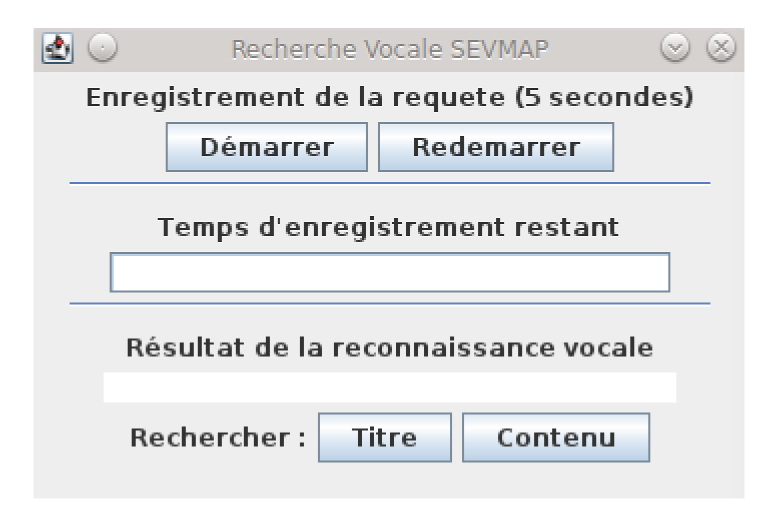
Des extensions de langages informatique seront proposées

Une recherche vocale

Une recherche vocale est également disponible.
On sélectionne dans l'onglet Fichier, une nouvelle recherche Vocale.

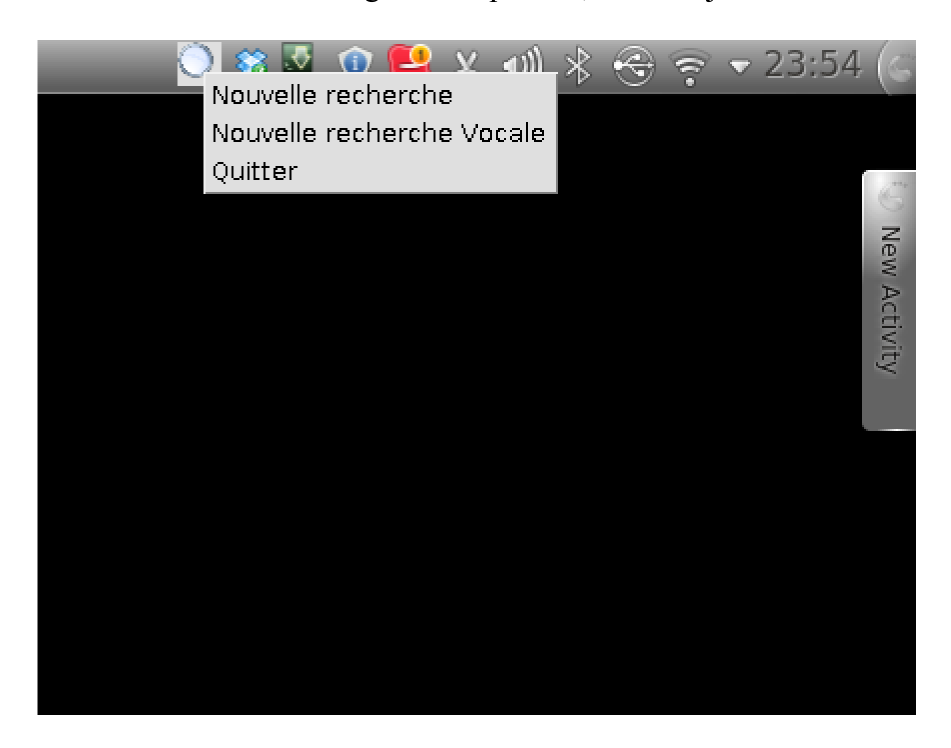


Voici à quoi ressemble la boite de dialogue :



On démarre l'enregistrement. A la fin des 5 secondes, le résultat de la requête s'affiche dans le cadre réservé. Cela permettra à l'utilisateur de corriger si l'enregistrement n'a pas réussi.

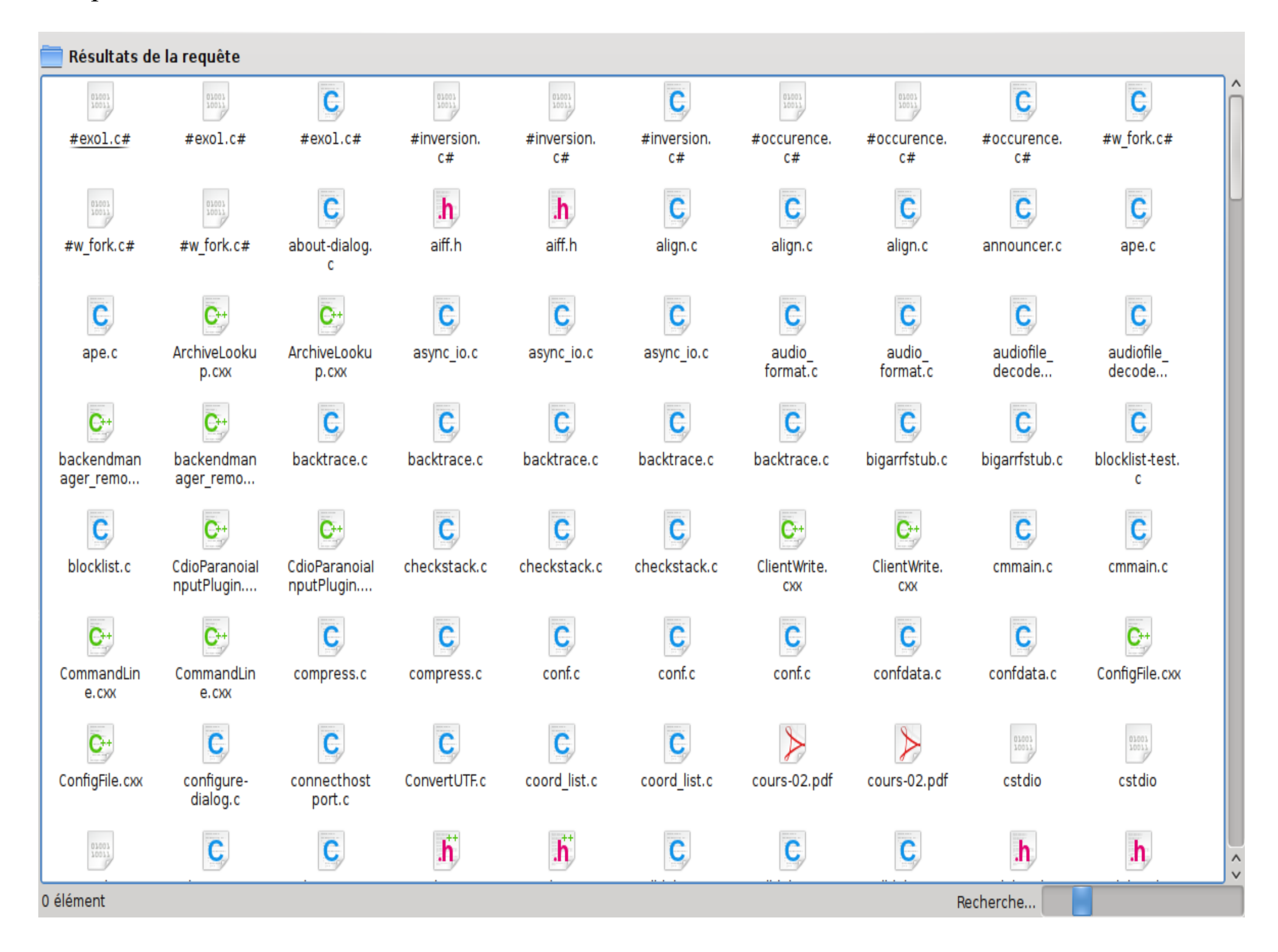
Enfin quelques raccourcis sont laissés à la disposition de l'utilisateur. Ils sont indiqués dans l'onglet Recherche, à côtés des options offertes.
Un raccourci bureau est également présent, il est toujours actif.



Affichage du résultat

Le résultat sera affiché dans une fenêtre semblable à l'explorateur de fenêtre utilisé par défaut sur la distribution GNU/Linux hôte.

Exemple : recherche de tous les fichiers contenant « *inclue* » :

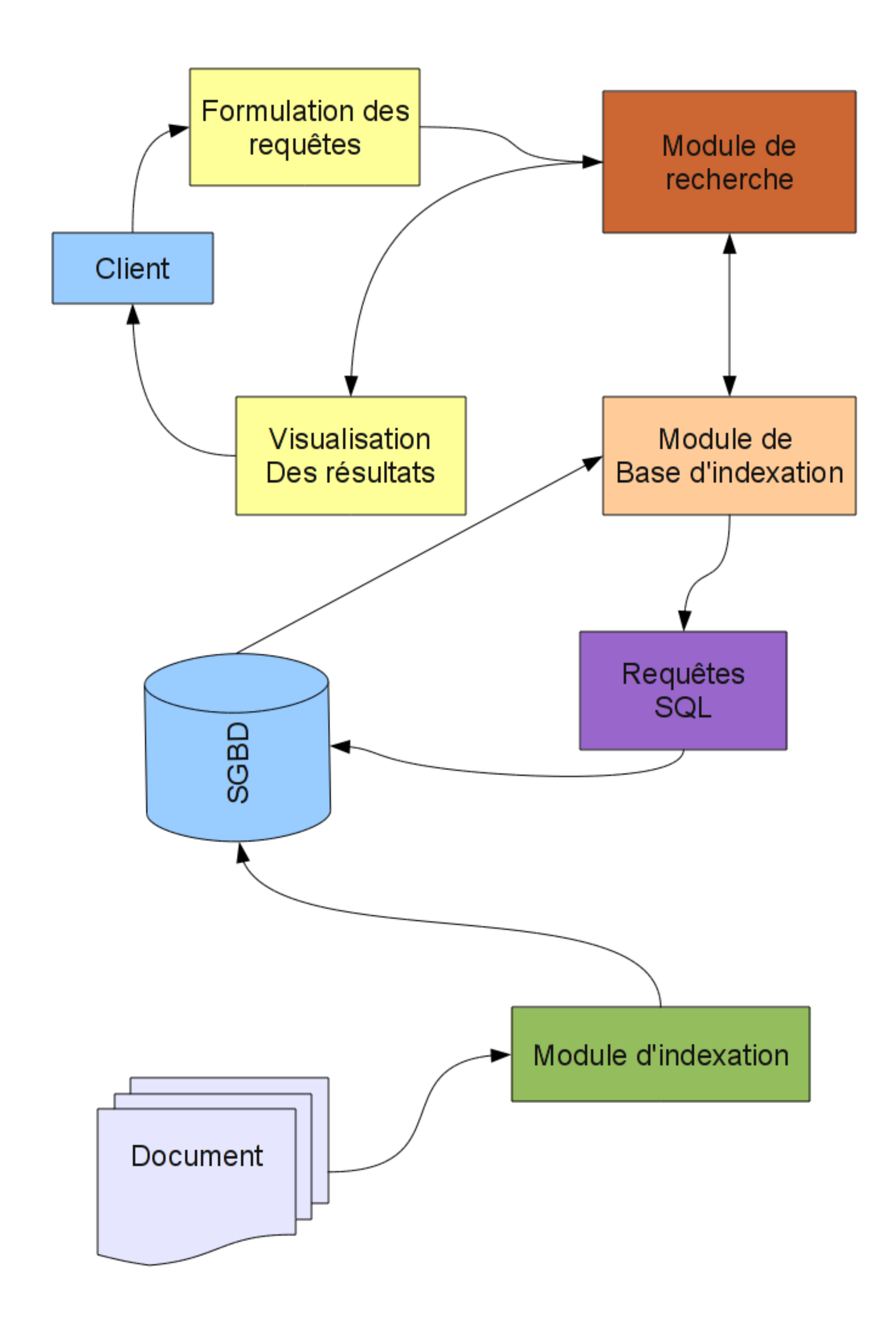


<u>Précision des résultats de recherche :</u>

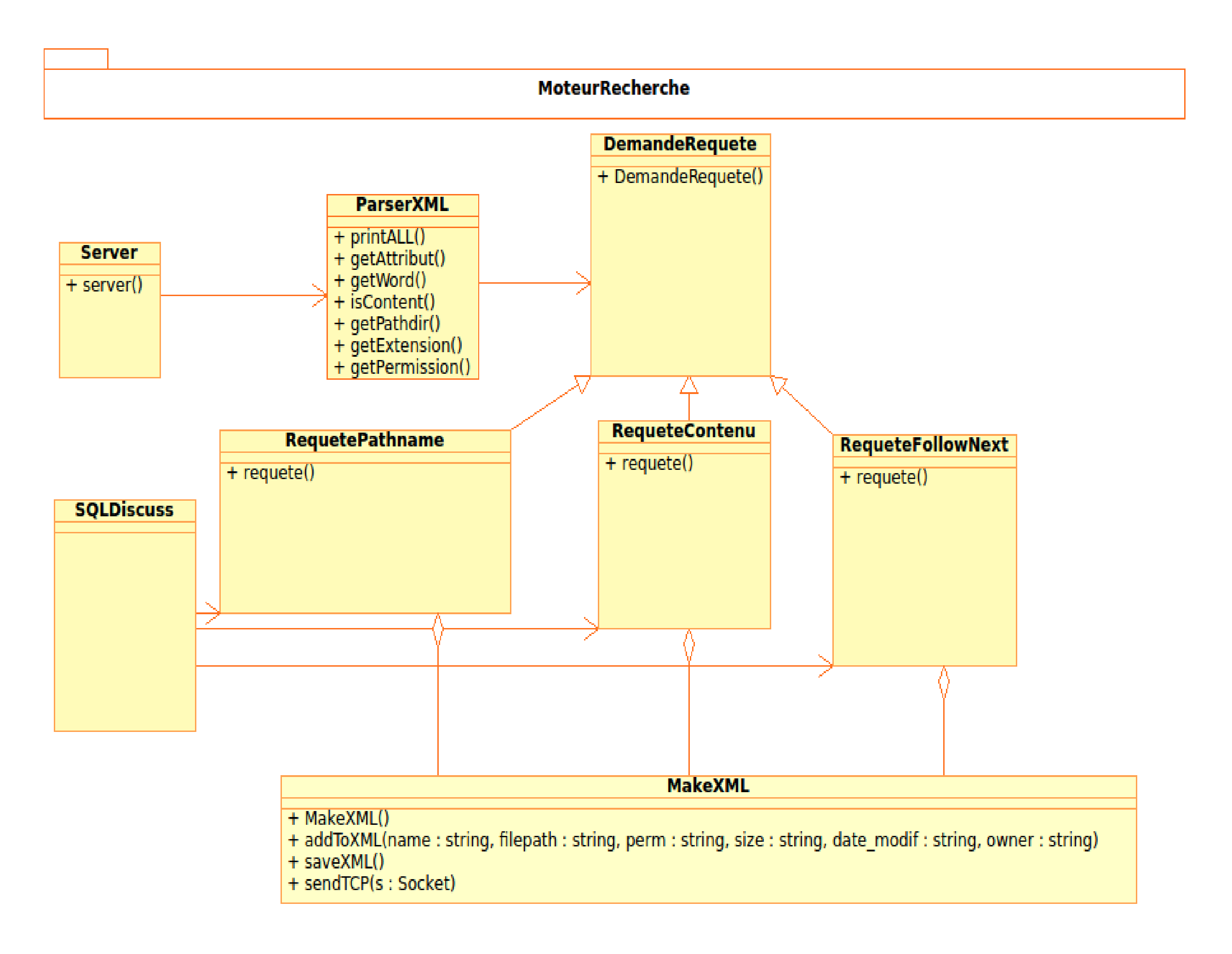
Les résultats de recherche sont affichés au fur et à mesure qu'ils sont trouvés par le moteur de recherche. Ainsi, si une recherche prend un temps non négligeable, l'utilisateur n'aura pas à attendre pour consulter les résultats. Les premiers d'entre eux seront affichés au bout de quelques secondes.

Descriptions des modules :

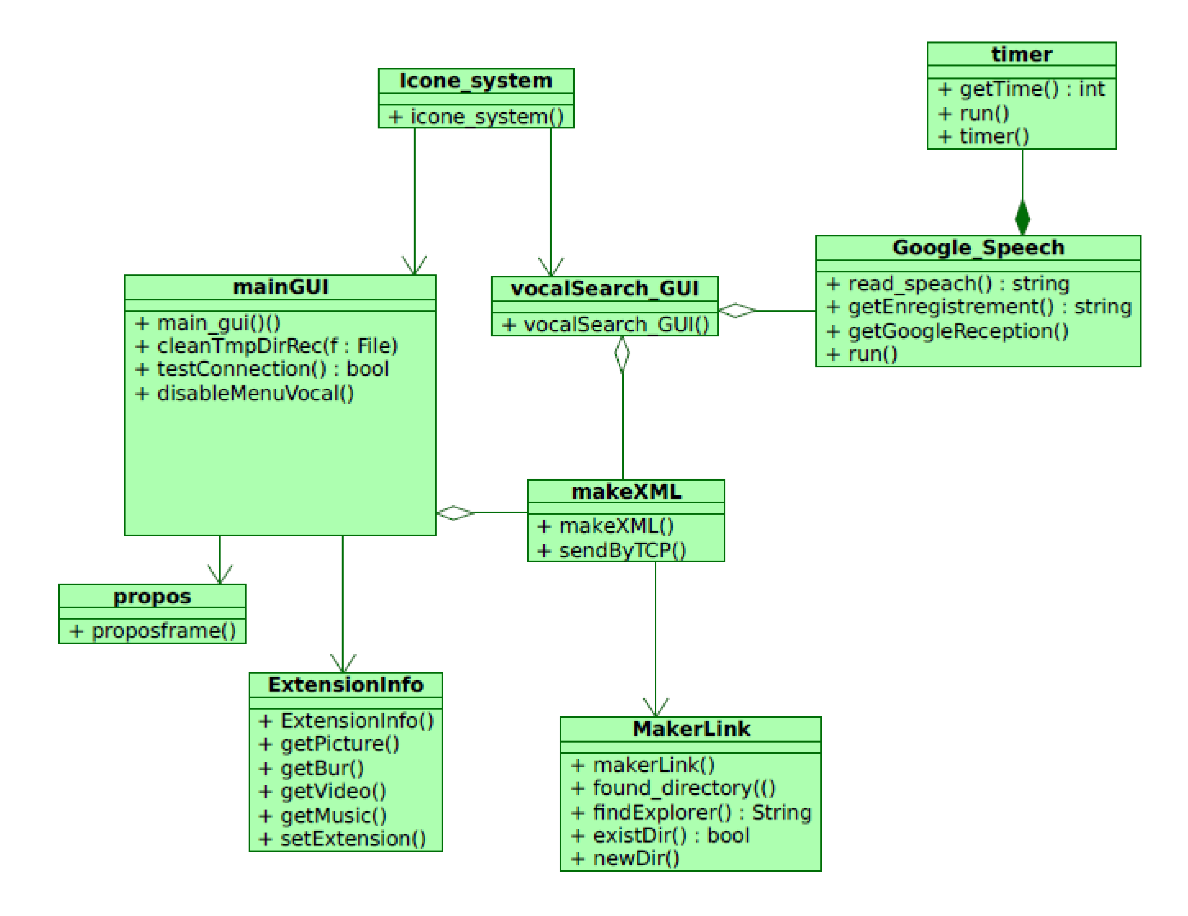
Schéma fonctionnel de l'application :



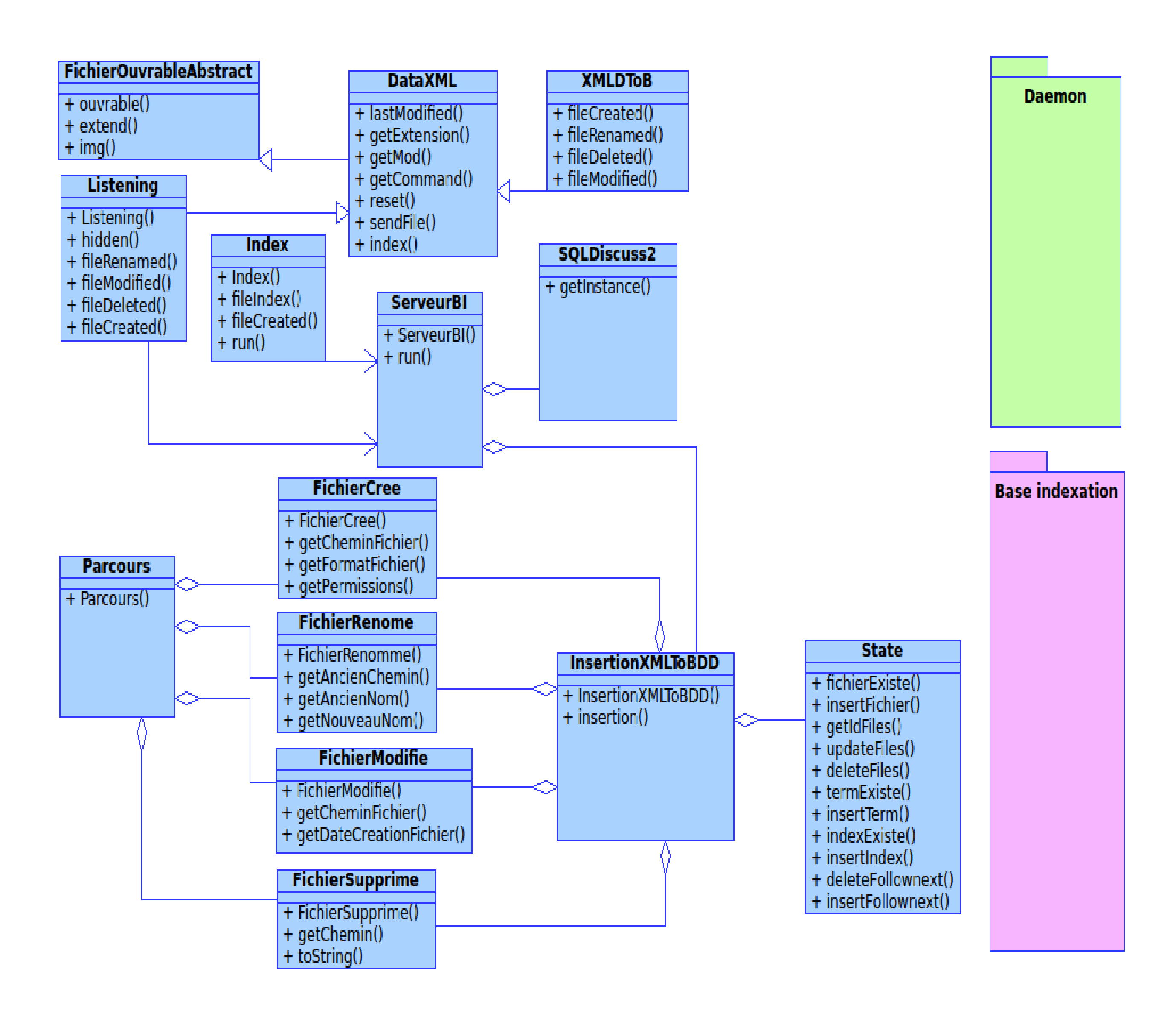
Diagrammes de classes



La classe Server est en attente d'un client. Lorsqu'un client se connecte, elle reçoit une suite de String correspondant au document xml, et appelle le parseur Xml pour récupérer toutes les informations contenues. ParserXml envoie les informations à la classe DemandeRequete qui contient donc les données que recherche l'utilisateur.
Le design pattern Fabrique : La classe DemandeRequete appelle de manière diffèrente les classes RequetePathname, RequeteContenu, RequeteFollownext selon la requête demandée. Puis ces classes lancent des requêtes à la base de données. Les données sont ensuite envoyées à la classe MakeXML qui les transmet à l'interface graphique.

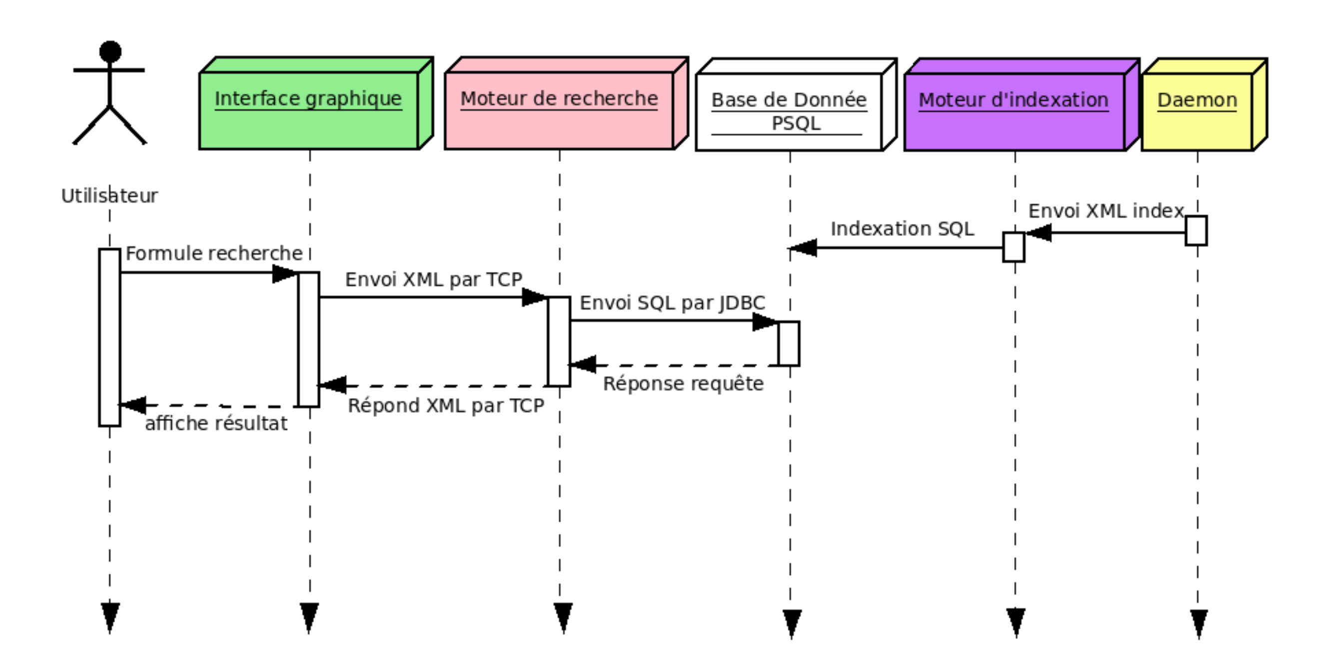


La classe icone_system correspond à l'icône du bureau, dans laquelle on peut appeler l'interface de recherche simple (Maingui) ou vocale (vocalSearch).
La classe Maingui contient le code de l'interface graphique de l'application.
La classe ExtensionInfo définit les différentes extensions proposées
La classe propos est une animation présentant l'équipe de programmeur.
La classe vocalSearch_GUI appelle la classe Google_Speech qui envoie sur google la requête vocale et récupère la réponse sous une chaine de caractère.
Des threads sont utilisées pour lancer simultanément l'enregistrement vocale et le timer (Classe Timer qui limite le temps d'une requête à 5sec).
makeXML et MakerLink ouvriront un explorateur de fichier et inséreront les résultats en liens symboliques.



FichierOuvrableAbstract est une classe abstraite qui contient deux méthodes pour connaître les fichiers ouvrables. La classe DataXML qui est aussi une classe abstraite hérite de la classe FichierOuvrable.
La classe Index fait le parcours du home et modifie le fichier Daemon.
La connexion à la base de donnée passe par la classe SQLDiscuss2.
La classe Listening ‘’écoute’’ sur le home, et envoie toute modification effectué à la classe ServeurBi.
La classe XMLTob vérifie et modifie le fichier Daemon. Elle est appelé par Listening.
La classe Parcours parcourt les fichiers et indexe chaque mot en lui associant sa liste de précédent et sa liste de suivant dans le fichier.
Les classes FichierCree, FichierRenomme, FichierModifie et FichierSupprime sont les classes représentant les fichiers envoyés par le démon.
La classe state envoie des requêtes à la base de données.

Diagramme de séquence présentant le chemin suivi d'une requête formulée par l'utilisateur :



Moteur de recherche :

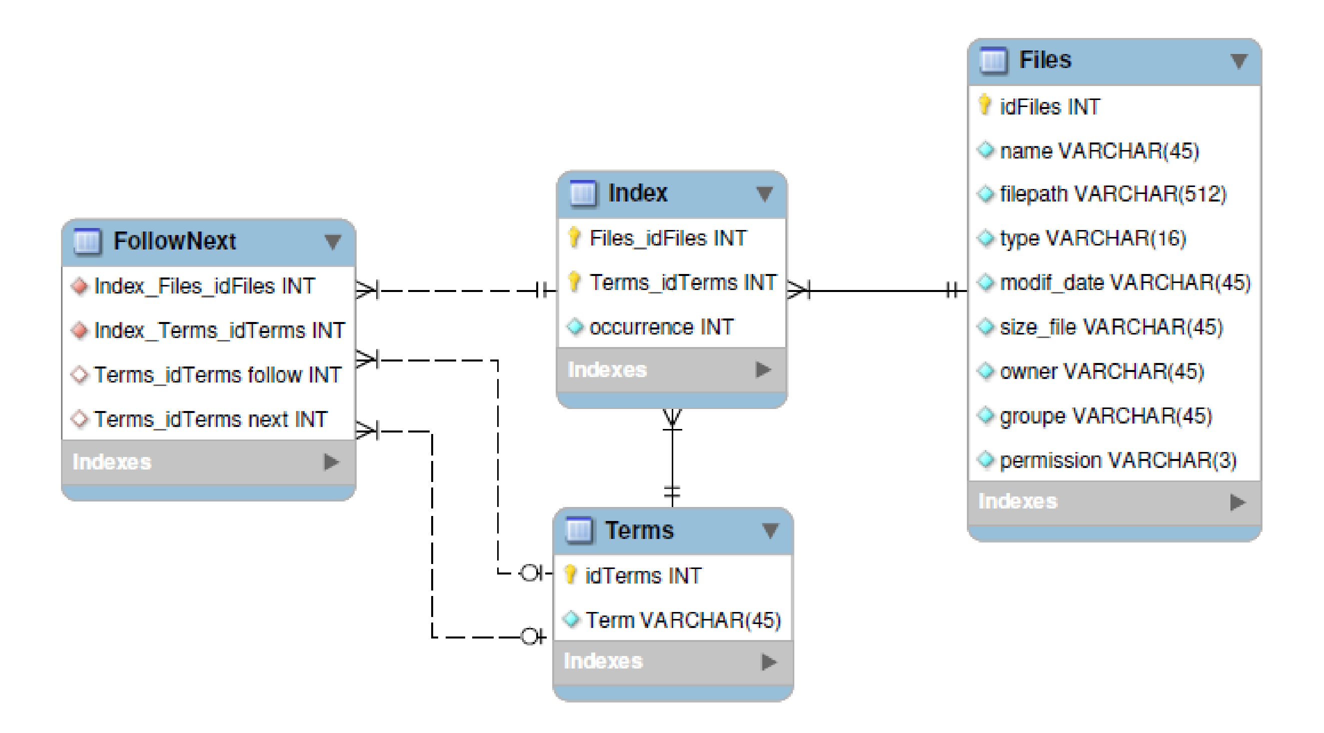
Ce module doit permettre de trier le résultat fourni par la base d'indexation. Il s'agit de conduire l'utilisateur vers la sélection des fichiers souhaités, via une liste des critères hiérarchiques, à savoir :

- nombre d'occurrence
- date de dernière modification
- nom
- suite de mots contenu dans le titre ou dans un fichier
- intervalle de temps donné
- enfin, la sélection de la période d'intérêt à partir d'un calendrier afin d'accéder à une période voulue plus rapidement à partir de 2 dates.

La base d'indexation :

Ce module permet de faire la transition entre le MR et le MI. Il stocke les données que le MI lui envoie dans un SGBD. Ce système permet de trier les données que le MI lui envoie, ce qui lui permet de faire une recherche plus efficace des requêtes demandées par le MR.

Conception de la base de données :



La base de données est composée de quatre tables.

La table *Terms* regroupe la liste de tous les termes que l'on peut retrouver dans l'indexation. Chaque terme est unique dans la table et est associé à un identifiant unique aussi, qui est clé primaire.

La table *Files* regroupe la liste de tous les fichiers indexés; chaque ligne de la table est composée d'un identifiant unique et d'une clé primaire, du nom du fichier, du nom du dossier contenant ce fichier, de son type (ex : .txt, .jpg, …), de sa date de création et de sa date de dernière modification.

La table FollowNext regroupe pour chaque terme, un couple de précédent et de suivant. Un attribut idFiles complétera ce triplet. Il est possible qu'un triplet se répète car il peut apparaître dans différents fichiers.

La table *Index* associe un fichier à ses termes. On y retrouve donc un identifiant de terme avec un identifiant de fichier. Chaque identifiant pouvant être présent plusieurs fois dans la table associée à un autre terme ou fichier.

Moteur d'indexation

L’outil d'indexation doit remplir la BI de façon intelligente. Par exemple, 10go de données ne doit pas créer une base de données d'une dizaine de Giga-octets. Une implémentation d’un algorithme de remplissage efficace et la plus condensée possible est nécessaire.

Indexation

Une particularité de l’indexation utilisée est qu’elle prendra pour chaque terme, apparaissant dans les fichiers, sa liste de précédents et de suivants. Ceci sera utile pour une recherche à plusieurs termes.

Indexation inversé

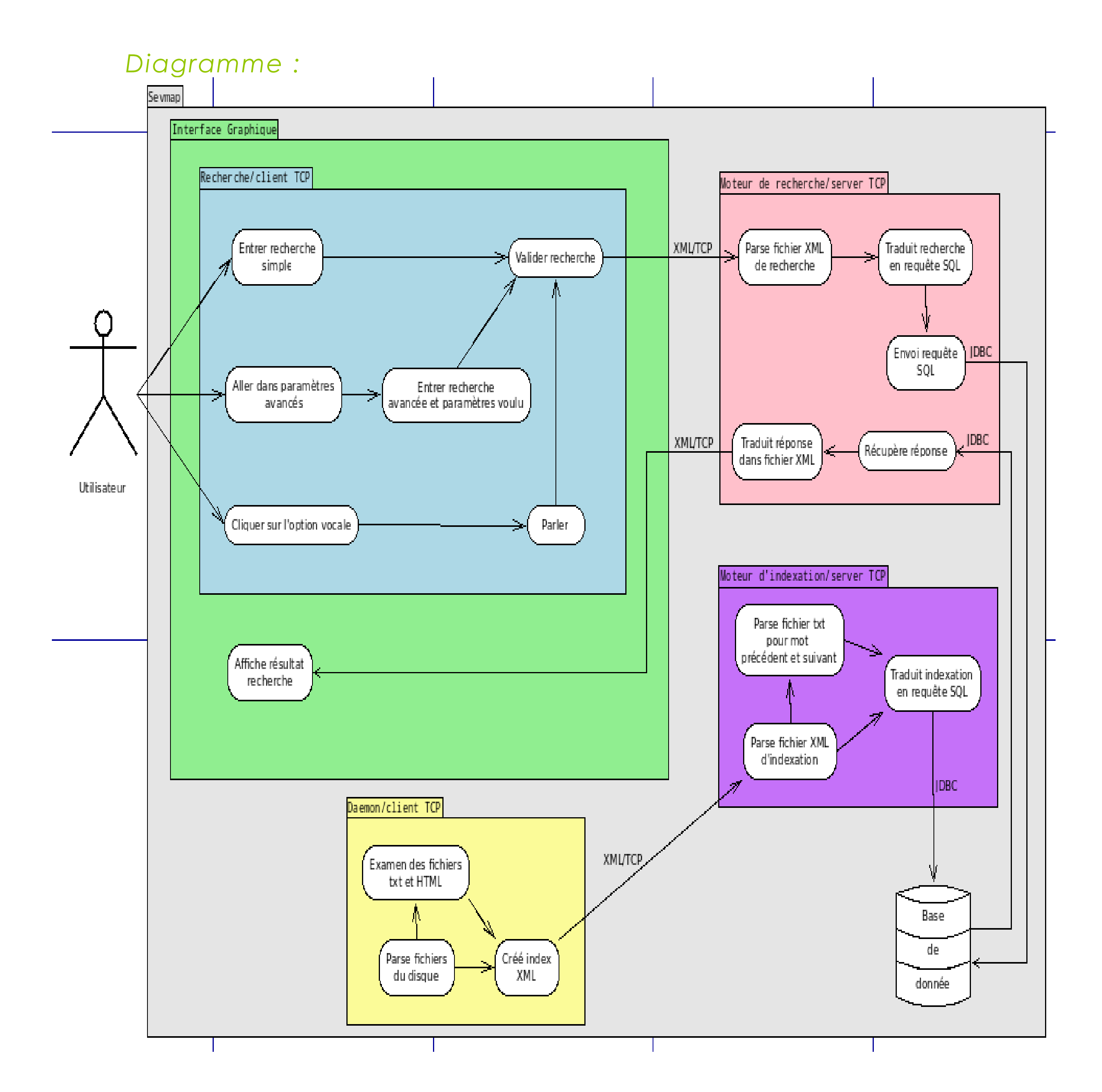
L'indexation des ressources récupérées consiste à extraire les mots considérés comme significatifs du corpus à explorer. Les mots extraits sont enregistrés dans une base de données organisée comme un gigantesque dictionnaire inverse. A titre de comparaison, l'index terminologique d'un ouvrage permet de retrouver rapidement dans quel chapitre se situe un terme significatif donné. Les termes non significatifs s'appellent des mots vides.

Types de champs

Les champs peuvent être stockés, et dans ce cas, leurs textes sont stockés dans l'index de la lettre d'une manière non inversée. Les champs qui sont inversés sont appelés indexés. Un champ peut être à la fois stocké et indexé.

Use Case :

Diagramme :



Cas 1:

Titre du cas d'utilisation:
Afficher les résultats d'une requête simple.
Acteurs qui participent au cas d'utilisation:
L'utilisateur.
Pré - conditions à l'exécution du cas d'utilisation:
L'utilisateur formule une requête dans la barre de recherche.
Le système est accessible.
Déclencheur du cas d'utilisation:
L'utilisateur clique sur le bouton "rechercher".

Scénario nominal:

1. L'utilisateur entre une requête dans la barre de recherche.
2. L'utilisateur valide la saisie en cliquant sur le bouton "rechercher".
3. Le MR envoie la requête formulée en format xml à la BI.
4. La BI récupère les fichiers correspondant à la requête.
5. La BI envoie les données au MR en format xml.
6. Le MR trie (range par pertinence, nombre d'occurrence de la requête) les données reçues.
7. Le MR envoie les données à l'interface.
8. L'interface affiche les résultats.

Scénario alternatifs:
a. Si la requête envoyée ne fournit aucun résultat.
a.1. L'interface informe l'utilisateur qu'aucun document n'a été trouvé.
b. Si la requête n'est pas valide.
b.1. L'interface informe l'utilisateur que la recherche n'a pu aboutir.

Post-conditions:

Affiche les documents.

Cas 2:

Titre du cas d'utilisation:
Afficher les résultats d'une requête « avancée ».
Acteurs qui participent au cas d'utilisation:
L'utilisateur.
Pré - conditions à l'exécution du cas d'utilisation:
L'utilisateur clique sur les paramètres avancés.
L'utilisateur peut formuler une requête dans la barre définie.
Le système est accessible.
Déclencheur du cas d'utilisation
1. L'utilisateur clique sur les paramètres avancés.
2. L'utilisateur clique sur le bouton "rechercher".

Scénario nominal:

1. L'utilisateur clique sur les paramètres avancés.
1.1. L'utilisateur peut formuler une requête.
2. L'utilisateur valide la saisie en cliquant sur le bouton "rechercher".
3. Le MR envoie la requête formulée en format xml à la BI.
4. La BI récupère les fichiers correspondant à la requête.
5. La BI envoie les données au MR en format xml.
6. Le MR trie (range par pertinence si précisé par l'utilisateur, sinon par occurrence) les données reçues.
7. Le MR envoie les données à l'interface.
8. L'interface affiche les résultats.

Scénario alternatifs:
a. Si la requête envoyée ne fournit aucun résultat.
a.1. L'interface informe l'utilisateur qu'aucun document n'a été trouvé.
b. Si la requête n'est pas valide.
b.1. L'interface informe l'utilisateur que la recherche n'a pu aboutir.

Post-conditions:

Affiche les documents.

Cas 3:

Titre du cas d'utilisation:
Afficher les résultats d'une requête vocale.
Acteurs qui participent au cas d'utilisation:
L'utilisateur.
Pré - conditions à l'exécution du cas d'utilisation:
L'utilisateur formule une requête vocale.
Le système est accessible.
Déclencheur du cas d'utilisation:
L'utilisateur clique sur le bouton "démarrer" et choisis le cas de recherche ‘titre’ ou ‘contenu’.

Scénario nominal:

1. L'utilisateur ouvre la recherche vocale dans l’onglet Fichier.
2. L'utilisateur appuie sur le bouton ‘’démarrer’’.
3. L’utilisateur formule à haute voix sa recherche de durée maximum 5sec.
4. Le résultat est affiché dans la fenêtre.
5. L’utilisateur lance la recherche.
3. Le MR envoie la requête formulée en format xml à la BI.
4. La BI récupère les fichiers correspondant à la requête.
5. La BI envoie les données au MR en format xml.
6. Le MR trie (range par pertinence, nombre d'occurrence de la requête) les données reçues.
7. Le MR envoie les données à l'interface.
8. L'interface affiche les résultats.

Scénario alternatifs:
a. Si la requête envoyée ne fournit aucun résultat.
a.1. L'interface informe l'utilisateur qu'aucun document n'a été trouvé.
b. Si la requête n'est pas valide.
b.1. L'interface informe l'utilisateur que la recherche n'a pu aboutir.

Post-conditions:

Affiche les documents.

Besoin organisationnel

Développement de l'application

Environnement de développement :

- emacs.
- Eclipse.

Environnement d’exécution :

- stations de l’école sous GNU Linux, Mac OSX

La partie base de données de l’application sera développée en PostgreSQL. Les langages utilisés seront les langages Java et SQL.

Livraison

Une version déboguée et fonctionnelle devra être disponible pour le 21 Mai 2013.

Implantation

Standard

Besoin externe

Interopérabilité

Description du protocole
Comme on peut le voir dans le diagramme de séquence.
Le démon envoie un fichier Xml par TCP au moteur d'indexation. L'indexation est ensuite envoyé à la base de données.
Un fichier Xml est envoyé depuis l'interface au moteur de recherche. Une requête est envoyée à la base de donnée. La réponse à la requête sera renvoyé sous format XML en TCP à l'interface graphique.
Ces fichiers Xml servent à l'interopérabilité.

Éthique

Cette application sera disponible sous licence GPL v3.0, elle répondra aux quatre libertés fondamentales du logiciel libre.

- la liberté d'utiliser le logiciel
- la liberté de copier le logiciel
- la liberté d'étudier le logiciel
- la liberté de modifier le logiciel et de redistribuer les versions modifiées

Légalité

Nous implémenterons le code dans son ensemble. Aucune copie de code existant ne sera utilisée. Nous n'aurons donc aucun souci de copyright, ni de redevance quelconque.

Le logiciel ainsi que tous les outils utilisés, pour sa réalisation, seront disponibles.

Respect de la vie privée

L'application n'enverra aucune donnée sur des serveurs externes. Le respect de la vie privée sera donc garanti.

Evolution de l'application

L'application pourra être améliorée en intégrant par exemple :

– Une connexion à distance permettant de connaître le contenu de la machine distante.

ANNEXES

Glossaire :

Environnement de développement : logiciel permettant d'écrire l'application dans un langage déterminé.

JAVA : langage de programmation crée par Sun Microsystems.

SQL : langage informatique normalisé servant à effectuer des opérations sur des bases de données relationnelles.

SGBD : Acronyme de Système de Gestion de Base de Données destiné à stocker et à partager des informations dans une base de données.

Postgresql : système de gestion de base de données relationnelle.

XML : Acronyme d'Extensible Markup Language, est un langage informatique de balisage générique. Celui-ci permet de faciliter l'échange automatisé de contenus complexes entre systèmes d'informations hétérogènes.

DTD : Acronyme de Document Type Definition, est un document permettant de décrire un modèle de document XML.

Composition de l'équipe :

Nom d'équipe : SEVMAP

Stéphane WONG : Elève de master à l'université Paris-Diderot parcours Système Réseaux et Internet, chef d'équipe.

Enis SULJOVIC : Elève de master à l'université Paris-Diderot parcours Système Réseaux et Internet.

Victor OUDIN : Elève de master à l'université Paris-Diderot parcours libre.

Marc-Anthony RICHARD : Elève de master à l'université Paris-Diderot parcours Système Réseaux et Internet,

Ali BEN EL HADJ : Elève de master à l'université Paris-Diderot parcours Système Réseaux et Internet.

Pierre DARGHAM : Elève de master à l'université Paris-Diderot parcours langage de programmation.